Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 10–16

BILEVEL DERIVATIVE-FREE OPTIMIZATION AND ITS APPLICATION TO ROBUST OPTIMIZATION

A. R. CONN AND L. N. VICENTE

Abstract: We address bilevel programming problems when the derivatives of both the upper and the lower level objective functions are unavailable.

The core algorithms used for both levels are trust-region interpolation-based methods, using minimum Frobenius norm quadratic models when the number of points is smaller than the number of basis components. We take advantage of the problem structure to derive conditions (related to the global convergence theory of the underlying trust-region methods, as far as possible) under which the lower level can be solved inexactly and sample points can be reused for model building. In addition, we indicate numerically how effective these expedients can be. A number of other issues are also discussed, from the extension to linearly constrained problems to the use of surrogate models for the lower level response.

One important application of our work appears in the robust optimization of simulation-based functions, which may arise due to implementation variables or uncertain parameters. The robust counterpart of an optimization problem without derivatives falls in the category of the bilevel problems under consideration here. We provide numerical illustrations of the application of our algorithmic framework to such robust optimization examples.

Keywords: Bilevel programming, derivative-free optimization (DFO), robust optimization, simulation-based optimization, trust-region methods, quadratic interpolation.
AMS Subject Classification (2010): 90C30, 90C47, 90C56.

1. Introduction

In this paper we address bilevel problems of the form

$$\begin{aligned} \min_{(x^u,x^\ell)\in\mathbb{R}^{n^u\times n^\ell}} \quad & f^u(x^u,x^\ell) \\ \text{s.t.} \quad & x^\ell \in \arg\min \left\{ f^\ell(x^u,z^\ell) : \ z^\ell \in \mathbb{R}^{n^\ell} \right\}. \end{aligned} \tag{1}$$

where $f^u, f^\ell : \mathbb{R}^{n^u\times n^\ell} \to \mathbb{R}$ are the upper level and lower level objective functions, respectively. In a similar way, the variables x^u and x^ℓ are called the upper level and lower level variables. This paper addresses derivative-free unconstrained bilevel problems, i.e., problems of the type (1) in the context

Date: June 1, 2010.
Support for this work was provided by FCT under the grant PTDC/MAT/098214/2008.

where only function values for f^u and f^ℓ are available. We refer the reader to the books on bilevel optimization by Bard [1], Dempe [10], and Shimizu, Ishizuka, and Bard [16] for theory and algorithms about more general classes of bilevel programming.

If we define the set of lower level minimizers, for a given x^u, by

$$x^\ell(x^u) = \arg\min \left\{ f^\ell(x^u, z^\ell) : \ z^\ell \in \mathbb{R}^{n^\ell} \right\},$$

and assume that this set is a singleton, we can rewrite the bilevel problem, equivalently, using only the upper level variables

$$\min_{x^u \in \mathbb{R}^{n^u}} \ f^u(x^u, x^\ell(x^u)). \tag{2}$$

In an analogy to the nomenclature used in optimal control, we refer to formulation (2) as the reduced formulation of the original problem (1). We will call $f^u(x^u, x^\ell(x^u))$ the reduced upper level function.

An alternative formulation is derived using both upper and lower level variables. In fact, if the lower level problem is convex and continuously differentiable in the lower level variables, the bilevel problem is equivalent to

$$\begin{aligned} \min_{(x^u, x^\ell) \in \mathbb{R}^{n^u \times n^\ell}} \quad & f^u(x^u, x^\ell) \\ \text{s.t.} \quad & \nabla_\ell f^\ell(x^u, x^\ell) = 0, \end{aligned} \tag{3}$$

where $\nabla_\ell f^\ell$ designates the partial gradient of f^ℓ with respect to x^ℓ. Under no convexity assumption for the lower level problem, the feasible set of problem (3) is only guaranteed to be contained in the feasible set of the original problem (1) and therefore the optimal value of (3) is only guaranteed to be a lower bound for the optimal value of (1), if both optimal values exist. The formulation (3) is called, again using the same comparison to optimal control, the all-at-once formulation. In this paper, we chose to work with the reduced formulation (2) rather than the all-at-once formulation (3). Given the derivative-free, possibly nonconvex context in which the lower level problem is posed, we felt this was a more direct place to begin.

We do not address bilevel problems with general (upper level or lower level) constraints. One of the reasons being that a good understanding of the unconstrained case seems to be necessary to tackle the constrained one. Also, as it can be seen from the current paper, derivative-free unconstrained bilevel optimization already introduces a number of nontrivial issues and a considerable amount of technical detail. However, as we will see, linear

upper and lower level constraints (and nonlinearities in terms of the upper level variables in the lower level constraints) can be easily incorporated in our approach.

In Sections 2 and 3, we review some aspects of trust-region methods required for this paper. Using known facts related to the global convergence of these methods and error bounds for polynomial interpolation of perturbed functions, we show in Section 4 how to develop conditions to inexactly solve the lower level problem and reuse points at which the lower level function has been previously evaluated. An interpolation-based trust-region framework is developed in Section 5 for the derivative-free solution of bilevel optimization problems, incorporating the techniques for inexactness and re-usage of points previously mentioned. We will show in Section 6 how effective these techniques can be in terms of saving lower level function evaluations. In Sections 5 and 6 we also address issues like the use of surrogate models for the lower level response and the extension to linearly constrained problems. The application of our algorithmic framework to derivative-free robust optimization problems is discussed in Section 7. The paper is concluded in Section 8 with some final remarks.

2. Trust-region interpolation-based models for DFO

In this section and in the next we consider the unconstrained minimization of a smooth function $f : \mathbb{R}^n \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^n} \; f(x).$$

In addition, assume the existence of a quadratic model $m(x+s)$ for f around a point x,

$$m(x+s) \;=\; f(x) + \langle g, s\rangle + \frac{1}{2}\langle s, Hs\rangle,$$

where Δ is the trust-region radius and g and H are some approximations to, respectively, the gradient and Hessian of f at x. The algorithmic framework described later is based on the trust-region technique, where the step s is computed by approximately solving the trust-region subproblem

$$\min_{s \in B(0;\Delta)} \; m(x+s). \tag{4}$$

Typically, trial points are accepted in trust-region methods when the ratio between the actual reduction in the function and the reduction predicted by

the model,

$$\rho \;=\; \frac{f(x) - f(x+s)}{m(x) - m(x+s)},$$

is above a positive threshold η_0. In this case, one achieves a sufficient decrease in the function, in the sense that $f(x) - f(x+s) > \eta_0(m(x) - m(x+s))$.

The design of trust-region methods based on interpolation models received some attention before the late nineties but mostly from a practical point of view (with original contributions from Winfield and Powell; see [9, Section 10.9]). The idea is simple and consists of building quadratic interpolation models $m(x+s)$ out of sample sets which are updated along the minimization process.

Later, Conn, Scheinberg, and Toint [6] introduced the so-called criticality step and designed and analyzed the first interpolation-based derivative-free trust-region method globally convergent to first-order critical points. The criticality step, taken at the beginning of each trust-region iteration, consists of improving the geometry of the sample set (and consequently of the interpolation model) for a possible smaller, pre-specified trust-region radius, whenever the model gradient gets sufficiently small.

Conn, Scheinberg, and Vicente [8] studied the appropriate incorporation of fully linear and fully quadratic models, global convergence when acceptance of iterates is based on simple decrease of the objective function ($f(x) - f(x+s) > 0$), and global convergence for second-order critical points. Due to the use of the criticality step, they proved that the trust-region radius converges to zero.

From a more practical point-of view, Powell has intensively studied these methods, bringing to the attention of the community the advantage of using quadratic models based on minimum Frobenius norm minimization [15] when the number of points is smaller than the number of basis components.

3. Trust-region interpolation-based models for DFO with inexact function values

In the context that we are interested, the function f cannot be evaluated exactly but the absolute error between the true function value and the inexact function value is controllable. Instead of $f(x)$ we compute $\bar{f}(x;\epsilon_x)$ and we can enforce, if we wish, that

$$|f(x) - \bar{f}(x;\epsilon_x)| \;\leq\; \epsilon_x. \tag{5}$$

The tolerance $\epsilon_x > 0$ associated with x might not be even specified when computing $\bar{f}(x;\epsilon_x)$. The point here is that we assume that if a tolerance ϵ_x is given we can compute an inexact $\bar{f}(x;\epsilon_x)$ corresponding to $f(x)$ with an absolute error satisfying (5).

The amount of inexactness allowed in the function calculation is measured by a fraction η_0 of the decrease $m(x) - m(x+s)$ predicted by the model (of $\bar{f}(x;\epsilon_x)$, now, rather than $f(x)$). Let ϵ and ϵ_{x+s} be upper bounds on the absolute values of the errors incurred while approximating $f(x)$ and $f(x+s)$, respectively. These errors must satisfy:

$$\max\{\epsilon_x, \epsilon_{x+s}\} \;\leq\; \eta_0' \left(m(x) - m(x+s)\right). \tag{6}$$

It can be easily checked (see [5, Section 10.6]) that if the ratio ρ between actual and predicted reductions for the inexact function $\bar{f}$ satisfies

$$\frac{\bar{f}(x;\epsilon_x) - \bar{f}(x+s;\epsilon_{x+s})}{m(x) - m(x+s)} \;\geq\; \eta_0$$

then

$$\frac{f(x) - f(x+s)}{m(x) - m(x+s)} \;\geq\; \eta_0' - 2\eta_0 \;>\; 0,$$

with $0 < \eta_0 < \eta_0'/2$ and $\eta_0 < 1$.

4. How to reduce the lower level solution effort

We start by stating two results on errors bounds for quadratic interpolation models when the functions being interpolated are subject to error. The first result ensures that such models, when using $(n+1)/(n+2)/2$ interpolation points, can exhibit the accuracy of fully quadratic models (in the sense of [9, Definition 6.2]) and requires the following assumption.

Assumption 4.1. *We assume that $Y = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$, with $p_1 = p+1 = (n+1)(n+2)/2$, is a poised set of sample points (in the quadratic interpolation sense, $d = 2$) contained in the (smallest possible) ball $B(y^0; \Delta(Y))$ of radius $\Delta = \Delta(Y)$.*

Further, we assume that the function f is twice continuously differentiable in an open domain Ω containing $B(y^0;\Delta)$ and $\nabla^2 f$ is Lipschitz continuous in Ω with constant $\nu_2 > 0$.

The proof of the following result is essentially an adaptation of the proof of [7, Theorem 3.16] (see also [9, Theorem 5.4]).

Theorem 4.1. *Let Assumption 4.1 hold. Assume that the error in function values is of the order of $\Delta(Y)^3$. Then, for all points y in $B(y^0;\Delta(Y))$, we have that*

- *the error between the Hessian of the quadratic interpolation model and the Hessian of the function satisfies*

$$\|\nabla^2 f(y) - \nabla^2 m(y)\| \ \leq \ \kappa_{eh}\,\Delta,$$

- *the error between the gradient of the quadratic interpolation model and the gradient of the function satisfies*

$$\|\nabla f(y) - \nabla m(y)\| \ \leq \ \kappa_{eg}\,\Delta^2, \tag{7}$$

- *the error between the quadratic interpolation model and the function satisfies*

$$|f(y) - m(y)| \ \leq \ \kappa_{ef}\,\Delta^3,$$

where κ_{eh}, κ_{eg}, and κ_{ef} are positive constants depending, essentially, on the geometry of the sample set (i.e., on the corresponding Λ–poisedness constant) and the Lipschitz constants of the second-order derivatives of f.

The next result ensures that quadratic interpolation models built using less than $(n+1)/(n+2)/2$ points can exhibit the accuracy of fully linear models (in the sense of [9, Definition 6.2]) and is derived under the following assumption.

Assumption 4.2. *We assume that $Y = \{y^0, y^1, \ldots, y^p\} \subset \mathbb{R}^n$ is a set of sample points poised in the linear interpolation sense (or in the linear regression sense if $p > n$) contained in the (smallest possible) ball $B(y^0;\Delta(Y))$ of radius $\Delta = \Delta(Y)$.*

Further, we assume that the function f is continuously differentiable in an open domain Ω containing $B(y^0;\Delta)$ and ∇f is Lipschitz continuous in Ω with constant $\nu > 0$.

The following result can be derived by making the appropriate changes in [9, Theorem 5.4].

Theorem 4.2. *Let Assumption 4.2 hold. Assume that the error in function values is of the order of $\Delta(Y)^2$. Then, for all points y in $B(y^0;\Delta(Y))$, we have that*

- *the error between the gradient of the quadratic interpolation model and the gradient of the function satisfies*

$$\|\nabla f(y) - \nabla m(y)\| \ \leq \ \kappa_{eg}\,\Delta,$$

- *the error between the quadratic interpolation model and the function satisfies*

$$|f(y) - m(y)| \ \leq \ \kappa_{ef}\,\Delta^2,$$

where κ_{eh}, κ_{eg}, and κ_{ef} are positive constants depending, essentially, on the norm of $\nabla^2 m(y)$, the geometry of the sample set (i.e., on the corresponding Λ–poisedness constant), and the Lipschitz constants of the first-order derivatives of f.

4.1. Inexact solution of the lower level problem. If the inexact solution $x^\ell_{dfo}(x^u)$ of the lower level problem is sufficiently close to the true solution $x^\ell(x^u)$ in the sense of the inverse function theorem (see, e.g., [14, Paragraph 5.2.1]), meaning that both points are in a neighborhood U where $\nabla_\ell f^\ell$ is continuously differentiable and $\nabla^2_{\ell\ell} f^\ell$ is nonsingular, then

$$\|x^\ell(x^u) - x^\ell_{dfo}(x^u)\| \ \leq \ c^\ell_\nabla \|\nabla_\ell f^\ell(x^u, x^\ell_{dfo}(x^u))\|,$$

where the constant c^ℓ_∇ depends essentially on the bound of the inverse of $\nabla^2_{\ell\ell} f^\ell$ in U. If (7) is valid for $f = f^\ell$ and $m = m^\ell$ at $(x^u, x^\ell_{dfo}(x^u))$, then (with $c^\ell_g = \kappa_{eg}$)

$$\|x^\ell(x^u) - x^\ell_{dfo}(x^u)\| \ \leq \ c^\ell_\nabla \|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\| + c^\ell_\nabla c^\ell_g (\Delta^\ell)^2.$$

Then,

$$\begin{aligned}
&|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| \\
&\leq \ \nu^u_f \|x^\ell(x^u) - x^\ell_{dfo}(x^u)\| \\
&\leq \ \nu^u_f c^\ell_\nabla \|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\| + \nu^u_f c^\ell_\nabla c^\ell_g (\Delta^\ell)^2.
\end{aligned}$$

4.1.1. *Interpolation requirements.* Thus, if

$$\|\nabla_\ell m^\ell(x^u, x^\ell_{dfo}(x^u))\| \ = \ \mathcal{O}((\Delta^u)^2)$$

and

$$\Delta^\ell \ = \ \mathcal{O}(\Delta^u),$$

then

$$|f^u(x^u, x^\ell(x^u)) - f^u(x^u, x^\ell_{dfo}(x^u))| = \mathcal{O}((\Delta^u)^2),$$

and this provide us a sufficient condition to solve inexactly the lower level problem whilst ensuring that the upper level model stays fully linear (see Theorem 4.2).

4.1.2. *Dynamic accuracy requirements.* One way to approximately enforce the dynamic accuracy requirement[1] (6) is to consider only $\epsilon_{x^u+s^u} \leq \eta_0'(m^u(x^u) - m^u(x^u+s^u))$ which, from above, is satisfied if

$$\|\nabla_\ell m^\ell(x^u+s^u, x^\ell_{dfo}(x^u+s^u))\| \;=\; \mathcal{O}\left(\min(\|s^u\|^2, \|s^u\|\|g^u\|)\right)$$

and

$$\Delta^\ell \;=\; \mathcal{O}\left(\sqrt{\min(\|s^u\|^2, \|s^u\|\|g^u\|)}\right).$$

4.2. Reusing previous (upper level perturbed) evaluated points. Especially close to the termination of the overall optimization process, when the upper level variables change little, it might be advantageous to use, in the solution of the lower level problem, points where the lower level function has been evaluated for slightly different values of the upper level variables. According to Theorem 4.1, the lower level model stays fully quadratic provided the difference in upper level variables is of the order of Δ^3. In fact, suppose that the lower level function f^ℓ has been evaluated at (x^u_{pert}, x^ℓ) and that we are interested in solving the lower level problem for x^u. Since

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;\leq\; \nu^u_f \|x^u - x^u_{pert}\|,$$

if

$$\|x^u - x^u_{pert}\| \;=\; \mathcal{O}((\Delta^\ell)^3),$$

then

$$|f^\ell(x^u, x^\ell) - f^\ell(x^u_{pert}, x^\ell)| \;=\; \mathcal{O}((\Delta^\ell)^3).$$

This provides us a criterion to decide whether to accept previously evaluated points in the building of the lower level model (see Theorem 4.1).

[1]Note that we are identifying f with $f^u(\cdot, x^\ell(\cdot))$, $\bar{f}$ with $f^u(\cdot, x^\ell_{dfo}(\cdot))$, m with m^u, x with x^u, and s with s^u.

5. A practical DFO algorithm for bilevel optimization

5.1. Description of a practical DFO algorithm. The algorithm described below follows some of the basic ideas of the algorithm in Fasano, Morales, and Nocedal [11]. The main similarity lies in the way the sample set is updated. When the iteration is successful, the new point always enters the sample set. However, unlike [11], we discard the sample point farthest away from the new iterate (rather than the sample point farthest away from the current iterate).

In the approach used in this paper we allow the algorithm to start with less points than those needed to build a determined quadratic model. Whenever there are less points than $p_{\max} = (n+1)(n+2)/2$, we use minimum Frobenius norm interpolation to build our models. This poses additional issues to those considered in [11], where $p_{\max}$ points are always used. For instance, until the cardinality of the sample set reaches $p_{\max}$, we never discard points from the sample set and always add new trial points independently of whether or not they are accepted as new iterates.

Another difference from [11] is that we discard points that are too far from the current iterate when the trust-region radius becomes small (this is a kind of weak criticality condition), hoping that the next iterations refill the sample set resulting in a similar effect as a criticality step. Thus, the cardinality of our sample set might fall below $p_{\min} = n + 1$, the number required to build fully linear models. In such situations, we never reduce the trust-region radius.

Algorithm 5.1 (A practical DFO algorithm)**.**
Step 0: Initialization.

> **Initial values.** *Select values for the constants* $\epsilon_g(= 10^{-5}) > 0$, $\delta(= 10^{-5}) > 0$, $0 < \eta_0(= 10^{-4}) < \eta_1(= 0.25) < 1$, $\eta_2(= 0.75) > \eta_1$, *and* $0 < \gamma_1(= 0.5) < 1 < \gamma_2(= 2)$. *Set* $p_{\min} = n + 1$ *and* $p_{\max} = (n + 1)(n + 2)/2$. *Set the initial trust radius* $\Delta_0(= 1) > 0$.
> **Initial sample set.** *Let the starting point* x_0 *be given. Select as an initial sample set* $Y_0 = \{x_0, x_0 \pm \Delta_0 e_i, i = 1, \ldots, n\}$.
> **Function evaluations.** *Evaluate the objective function at all* $y \in Y_0$.

Set $k = 0$.
Step 1: Model building.

Form a quadratic model $m(x_k+s)$ of the objective function from Y_k, using quadratic interpolation if $|Y_k| = p_{\max}$, and using minimum Frobenius norm quadratic interpolation if $|Y_k| < p_{\max}$.

Step 2: Stopping criteria.

Stop if $\|g_k\| \leq \epsilon_g$ or $\Delta_k \leq \delta$.

Step 3: Step calculation.

Compute a step s_k by solving (approximately) the trust-region subproblem $min_{s\in\mathbb{R}^n} m(x_k + s)$ s.t. $\|s\| \leq \Delta_k$.

Step 4: Function evaluation.

Evaluate the objective function at $x_k + s_k$.

Step 5: Selection of the next iterate and trust radius update.

*If $\rho_k < \eta_0$, reject the trial step, $x_{k+1} = x_k$, and reduce the trust-region radius if $|Y_k| \geq p_{\min}$, $\Delta_k = \gamma_1\Delta_k$ (***unsuccessful iteration***).*
*If $\rho_k \geq \eta_0$, accept the trial step $x_{k+1} = x_k+s_k$ (***successful iteration***).*
(Possibly decrease trust-region radius, $\Delta_k = \gamma_1\Delta_k$, if $\rho_k < \eta_1$ and $|Y_k| \geq p_{\min}$.)
Increase the trust-region radius, $\Delta_{k+1} = \gamma_2\Delta_k$, if $\rho_k > \eta_2$.

Step 6: Update the sample set.

If $|Y_k| = p_{\max}$, set $y_k^{out} \in \operatorname{argmax} \|y - x_{k+1}\|$ (break ties arbitrarily).
If the iteration was successful:
If $|Y_k| = p_{\max}$, $Y_{k+1} = Y_k \cup \{x_{k+1}\} \setminus \{y_k^{out}\}$.
If $|Y_k| < p_{\max}$, $Y_{k+1} = Y_k \cup \{x_{k+1}\}$.
If the iteration was unsuccessful:
If $|Y_k| = p_{\max}$, $Y_{k+1} = Y_k \cup \{x_k + s_k\} \setminus \{y_k^{out}\}$ if $\|(x_k + s_k) - x_k\| \leq \|y_k^{out} - x_k\|$.
If $|Y_k| < p_{\max}$, $Y_{k+1} = Y_k \cup \{x_k + s_k\}$.

Step 7: Model improvement.

When $\Delta_{k+1} < 10^{-3}$, discard from Y_{k+1} all the points outside $B(x_{k+1}; 100\Delta_{k+1})$

Increment k by 1 and return to Step 1.

5.2. Description of the algorithm (upper level). We are now in a position to describe our main derivative-free algorithm for bilevel optimization. Essentially, we will apply the algorithm described above to the minimization

of the reduced upper level function $f^u(x^u, x^\ell(x^u))$. A number of relevant issues need to be taken into consideration when evaluating this function, since it requires some, possibly inexact, solution of the lower level problem. This is addressed in the algorithm below and in the next section.

Another point of deviation from the basic Algorithm 5.1 is the computation of initial points for the initial lower level minimizations at Step 0. One possibility is to use whatever is supplied by the user. What we suggest below, however, does not require a user suggestion for the initial lower level variables. Instead, we build a linear surrogate model $x^\ell_m(x^u)$ for $x^\ell(x^u)$ (from setting to zero the gradient with respect to x^ℓ of a quadratic model of $f^\ell(x^u, x^\ell)$ in x^u and x^ℓ) and then plug in different values for x^u whenever needed.

Algorithm 5.2 (A practical DFO algorithm for bilevel optimization)**.**
Step 0: Initialization.

Initial values. *As in Algorithm 5.1 (the constants are* ϵ^u_g, δ^u, η^u_0, η^u_1, η^u_2, γ^u_1, γ^u_2, n_u, *and* Δ^u_0*).*
Initial sample set. *As in Algorithm 5.1 (the initial points is* x^u_0 *and the initial sample set is* Y^u_0*).*
Computing initial starting points for the lower level. *Let the lower level starting point* x^ℓ_0 *be given. Evaluate the lower level function at* $Y_0 = \{x_0, x_0 \pm \Delta e_i, i = 1, \ldots, n\}$*, for some* $\Delta > 0$*, with* $x_0 = (x^u_0, x^\ell_0)$ *and* $n = n^u + n^\ell$*. Form the minimum Frobenius quadratic model of* f^ℓ *(now considered as a function of both the upper and lower level variables) at* Y_0*. From this model, form the linear model* $x^\ell_m(x^u)$*.*
Function evaluations. *For all* $y^u \in Y^u_0$*, evaluate (possibly inexactly) the reduced upper level function* $f^u(y^u, x^\ell(y^u))$*:*

- *Solve the lower level problem by applying a similar algorithm starting at* $x^\ell_m(y^u)$ *(see next subsection). Let* $x^\ell_{dfo}(y^u)$ *be the result of this lower level optimization.*
- *Evaluate* f^u *at* $(y^u, x^\ell_{dfo}(y^u))$*.*

Set $k = 0$*.*
Step 1: Model building. *As in Algorithm 5.1 (the new model is* $m^u(x^u_k + s^u)$*).*
Step 2: Stopping criteria. *As in Algorithm 5.1.*
Step 3: Step calculation. *As in Algorithm 5.1 (the new step is* s^u_k*).*
Step 4: Function evaluation. *Evaluate (possibly inexactly) the reduced upper level function* $f^u(x^u_k + s^u_k, x^\ell(x^u_k + s^u_k))$*:*

Solve the lower level problem for $x_k^u+s_k^u$ by applying a similar algorithm (see next subsection). Let $x_{dfo}^\ell(x_k^u+s_k^u)$ be the result of this lower level optimization.
Evaluate f^u at $(x_k^u+s_k^u, x_{dfo}^\ell(x_k^u+s_k^u))$.

Step 5: Selection of the next iterate and trust radius update. *As in Algorithm 5.1 (the new iterate is x_{k+1}^u and the new trust radius is Δ_{k+1}^u).*
Step 6: Update the sample set. *As in Algorithm 5.1 (the new sample set is Y_{k+1}^u).*
Step 7: Model improvement. *As in Algorithm 5.1. Increment k by* 1 *and return to Step 1.*

5.3. Description of the algorithm (lower level). The derivative-free algorithm used for the lower level minimization is essentially the basic version (Algorithm 5.1) applied to $f^\ell(x^u,\cdot)$, with some differences that we describe below.

Let us consider first the case where the solution of the lower level problem has been requested for $x^u = x_k^u + s_k^u$ in Step 4 of Algorithm 5.2.

Reusing previous (upper level perturbed) evaluated points.
The initial lower level sample set is formed by first attempting to incorporate points (z_u^i, z_ℓ^i) for which the value of $f^\ell(z_u^i, z_\ell^i)$ has been computed and such that

$$\|z_u^i - (x_k^u + s_k^u)\| \;\leq\; \min(10^{-2}(\Delta_0^\ell)^3, 10^{-2}). \tag{8}$$

Then, if needed (i.e., if there are not $2n+1$ points in this situation), the rest of the sample set Y_0^ℓ is completed with points in $\{x_0^\ell, x_0^\ell \pm \Delta_0^\ell e_i, i = 1,\ldots,n^\ell\}$.

Inexact solution of the lower level problem.
The gradient model stopping tolerance is set to

$$\epsilon_g^\ell \;=\; \max(\min(10^{-2}\|s_k^u\|, 10^{-2}\|g_k^u\|^2, 10^{-2}), 10^{-5}). \tag{9}$$

Note that this meets the requirements for both reduced upper level interpolation and upper level dynamic accuracy (except for the size of the lower level trust-region radius) — see the previous section.

The lower level problem is also required to be solved for $x^u = y^u$, for all $y^u \in Y_0^u$, in Step 0 of Algorithm 5.2. In this case, $x_k^u + s_k^u$ is replaced by y^u in (8) and ϵ_g^ℓ is set to $\max(\min(10^{-2}\Delta_0^u, 10^{-2}), 10^{-5})$ in (9).

6. Implementation and examples

We have developed a relatively sophisticated Matlab implementation, along the lines described above (Algorithm 5.2 and Subsection 5.3), for the derivative-free solution of bilevel optimization problems.

6.1. The linearly constrained case. Although in principal the algorithmic approach followed in this paper could be generalized to more general constraints, the code we implemented handles bilevel problems with any type of linear constraints except upper level constraints on the lower level variables. In the reduced formulation, this latter type of constraints becomes nonlinear in the upper level variables and that poses an additional level of difficulty that we wish to avoid for the present.

More specifically, our code can handle bilevel optimization problems with linear constraints of the form:

$$\begin{aligned} \min_{(x^u,x^\ell)\in\mathbb{R}^{n^u\times n^\ell}} \quad & f^u(x^u,x^\ell) \\ & A^u x^u \le b^u, \\ & A^u_{eq} x^u = b^u_{eq}, \\ & \ell^u \le x^u \le u^u, \\ \text{s.t.} \quad & x^\ell \in \arg\min \big\{ f^\ell(x^u,z^\ell) : \; A^\ell z^\ell + B^\ell x^u \le b^\ell, \\ & \qquad\qquad A^\ell_{eq} z^\ell + B^\ell_{eq} x^u = b^\ell_{eq}, \\ & \qquad\qquad \ell^\ell + B^\ell_{bls} x^u \;\le\; z^\ell \;\le\; \ell^u + B^\ell_{bls} x^u \big\}, \end{aligned} \tag{10}$$

where $z^\ell \in \mathbb{R}^{n^\ell}$, $A^u \in \mathbb{R}^{m^u\times n^u}$, $b^u \in \mathbb{R}^{m^u}$, $A^u_{eq} \in \mathbb{R}^{m^u_{eq}\times n^u}$, $b^u_{eq} \in \mathbb{R}^{m^u_{eq}}$, $\ell^u \in (-\infty,\mathbb{R})^{n^u}$, $u^u \in (\mathbb{R},+\infty,)^{n^u}$, $\ell^u < u^u$, $A^\ell \in \mathbb{R}^{m^\ell\times n^\ell}$, $B^\ell \in \mathbb{R}^{m^\ell\times n^u}$, $b^\ell \in \mathbb{R}^{m^\ell}$, $A^\ell_{eq} \in \mathbb{R}^{m^\ell_{eq}\times n^\ell}$, $B^\ell_{eq} \in \mathbb{R}^{m^\ell_{eq}\times n^u}$, $b^\ell_{eq} \in \mathbb{R}^{m^\ell_{eq}}$, $\ell^\ell \in (-\infty,\mathbb{R})^{n^\ell}$, $u^\ell \in (\mathbb{R},+\infty,)^{n^\ell}$, $\ell^\ell < u^\ell$, and $B^\ell_{bls}, B^u_{bls} \in \mathbb{R}^{n^\ell\times n^u}$.

In fact, Algorithm 5.2 can be easily adapted to solve problems of the form (10) by essentially bringing a homogenous form of the linear constraints into the solution of the corresponding trust-region subproblems. In such cases, we change the shape of the trust-region constraints from ℓ_2 to ℓ_∞ to make the resulting trust-region subproblem a quadratic program. The lower level constraints $\ell^\ell + B^\ell_{bls} x^u \le z^\ell \le \ell^u + B^\ell_{bls} x^u$ are considered separately from inequality linear constraints since they still model simple bounds on the lower

level variables (despite the fact that the bounds depend on the values of the upper level variables).

Also, whenever the gradient of the model is used in Algorithm 5.2 as an indication of stationarity, we need to replace it by the projected gradient. Here, again, we measure distance in the most convenient norm. When only bounds are present, the ℓ_2-projection is computationally light. If there are only equality constraints, then the ℓ_2-projection can be computed by solving a linear system. In the more general cases, we replace the ℓ_2-norm by the ℓ_1 one in order to compute the projection from a linear, rather than a quadratic, program.

Note that our code could also handle any form of nonlinearity in terms of the upper level variables in the lower level constraints, since that would not change the linear form of those constraints in the lower level problem.

6.2. Some illustrative examples. We first tested the application of our derivative-free bilevel method (Algorithm 5.2) on two simple examples. In both cases $n^u = n^\ell$. The first example has no constraints and is defined by $f^u(x^u, x^\ell) = \sum_{i=1}^{n^u} (x_i^u)^2 + (x_i^\ell)^2$ and $f^\ell(x^u, x^\ell) = \|H_{\ell,\ell} x^\ell - x^u\|_2^4$, where $H_{\ell,\ell} = QDQ^\top$ with Q orthogonal and $D = \mathrm{diag}(1, \ldots, n^\ell)$. In the second one (taken from [4, Problem 4 in Appendix A4]), we have $f^u(x^u, x^\ell) = \sum_{i=1}^{n^u} x_i^\ell (x_i^u + 1)^2$, $f^\ell(x^u, x^\ell) = \sum_{i=1}^{n^\ell} x_i^\ell (x_i^\ell - x_i^u)$, $-2 \leq x_i^u \leq 1$, $i = 1, \ldots, n^u$ as upper level constraints, and $x_i^u \leq x_i^\ell$, $i = 1, \ldots, n^\ell$ as lower level constraints.

The result of the application of the algorithm to these two examples is described in Figure 1 when $n^u = n^\ell = 5$. For the first example, where the lower level function is quartic, it is apparent that the lower level effort reduction obtained by the techniques of Section 4 (detailed in Section 5), especially the inexact solution of the lower level problem, is significant. This effect is not so visible in the second example since the lower level function is quadratic (and thus mildly nonlinear) in the lower level variables. We ran our algorithm for larger instances of this problem ($n^u = n^\ell = 20$; see Figure 2) to give an indication of the overall effort needed to solve a larger problem.

7. Robust derivative-free optimization

In many real-world optimization problems, one is faced with data that is uncertain and one only has a representation of the problem determined by some form of estimation. When formulating such problems as robust optimization ones, immunization against data uncertainty is made by allowing

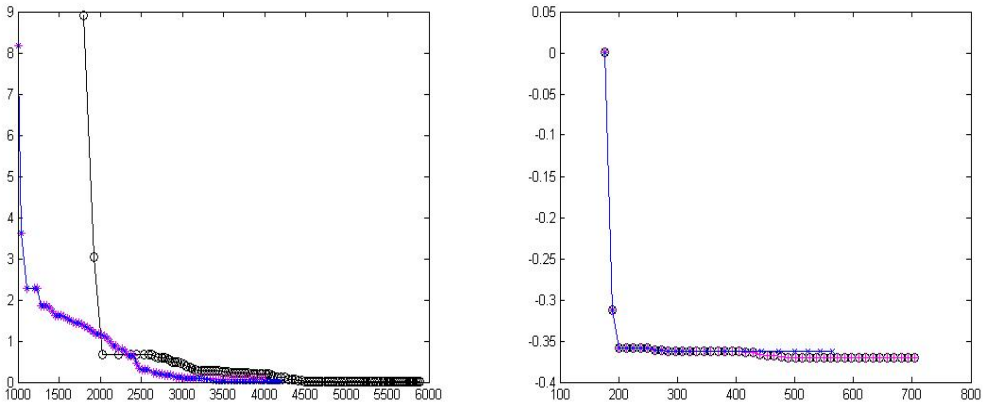

FIGURE 1. Result of the application of Algorithm 5.2 to the bilevel examples of Subsection 6.2 for $n^u = n^\ell = 5$. Both plots show the value of the reduced upper level function in terms of the number of lower level function evaluations. The black '$\circ$' curve correspond to the basic version. The red '$\times$' curve correspond to the version where the lower level problem is solved inexactly. The blue '$+$' curve correspond to the version where the lower level problem is solved inexactly and one reuses previous (upper level perturbed) evaluated points.

the uncertain parameters to become variables in uncertainty sets. One then looks for a safe, worst case scenario, hence the term robust. Robust optimization also provides a tool for dealing with parameters for which the optimal values must be later implemented in some practical context. Often such an implementation is error prone and, again, one would like to be protected against an unsuitable worst case scenario by a suitable choice of these parameters.

We are interested in the robust optimization of functions arising from simulation, where derivatives are difficult or expensive to compute. The function being minimized has the form $f(x, p)$, where x are traditional variables and p represents either an estimation of uncertain data or implementation parameters. We consider an uncertainty set P for the possible values of p. The robust optimization problem of interest to us can be then formulated as

$$\min_{(x,p)\in\mathbb{R}^n\times P} \max_{p\in P} f(x, p).$$

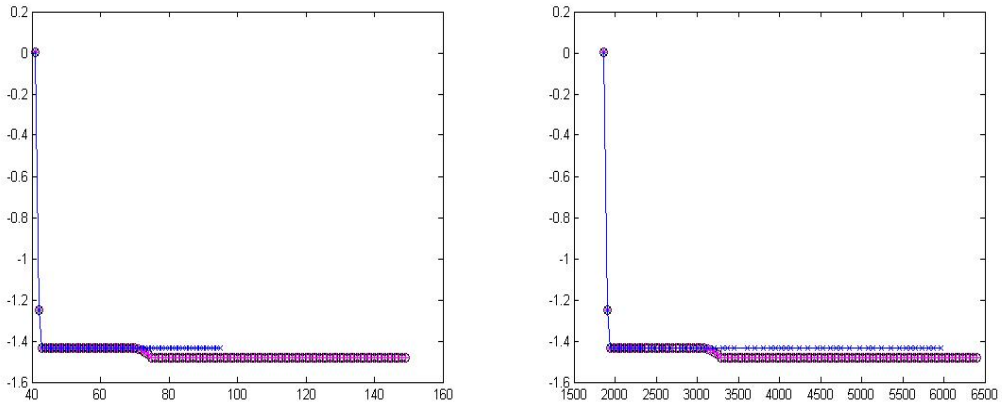

FIGURE 2. Result of the application of Algorithm 5.2 to the second bilevel example of Subsection 6.2 for $n^u = n^\ell = 20$. The plots show the value of the reduced upper level function in terms of the number of upper level function evaluations (left) and lower level function evaluations (right). (See the caption of Figure 1 for information about the curves.)

This problem can be reformulated as a bilevel optimization problem of the form

$$\begin{array}{ll} \min\limits_{(x,p)\in\mathbb{R}^n\times P} & f(x,p) \\ \text{s.t.} & p \in \arg\min\{-f(x,\bar{p}) : \bar{p} \in P\}. \end{array}$$

7.1. A simple example. We tested our algorithm in the example reported in [2]. In this example, the robust function is of the form $f(x,p) = g(x+p)$, where $x, p \in \mathbb{R}^2$ and

$$\begin{aligned} g(x) \;=\; & 2x_1^6 - 12.2x_1^5 + 21.2x_1^4 - 6.4x_1^3 - 4.7x_1^2 + 6.2x_1 \\ & + x_2^6 - 11x_2^5 + 43.3x_2^4 - 74.8x_2^3 + 56.9x_2^2 - 10x_2 \\ & - 0.1x_1^2x_2^2 + 0.4x_1^2x_2 + 0.4x_2^2x_1 - 4.1x_1x_2. \end{aligned}$$

The problem has one lower level constraint of the form $\|p\| \leq 0.5$ describing the set of possible implementation errors:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^2, p\in\mathbb{R}^2} & g(x+p) \\ \text{s.t.} & p \in \arg\min\left\{-g(x+\bar{p}) : \bar{p} \in \mathbb{R}^2, \|\bar{p}\| \leq 0.5\right\}. \end{array}$$

We ran our derivative-free bilevel method (Algorithm 5.2) for this problem starting from the same initial points as considered in [2] (namely point

$A = (-0.4 \quad 0.1)^\top$ and point $B = (1.8 \quad 2.4)^\top$). The results are plotted in Figure 3. In both cases, the algorithm identified a local minimizer of the original problem. The number of local minima of this robust optimization problem seems to be 5. We made a number of other tests for this small yet interesting problem. Convergence was observed in the vast majority of the cases to the closest local minimizer and was reasonably efficient in terms of the overall number of function evaluations.

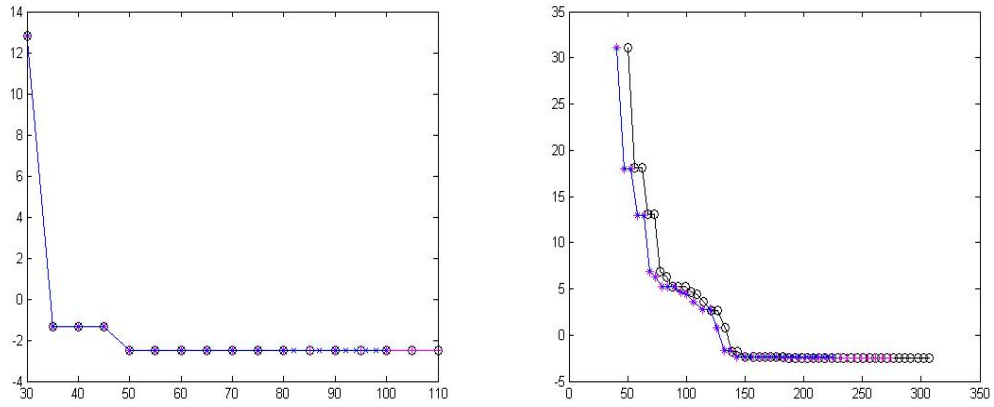

FIGURE 3. Result of the application of Algorithm 5.2 to the robust optimization example [2] (starting points A and B). Both plots show the value of the reduced upper level function in terms of the number of lower level function evaluations. (See the caption of Figure 1 for information about the curves.)

7.2. An example from portfolio optimization. Let the random variable R model the return of some asset or financial instrument. Consider a return level or threshold L. The performance of the asset can be measured by the so-called Omega function (introduced by Cascon, Keating, and Shadwick [3]), which is defined by the ratio of the weighted gains (above L) over the weighted losses (below L):

$$\Omega(R; L) \;=\; \frac{\int_L^{L_{max}} \mathbb{P}(R \geq r)dr}{\int_{L_{min}}^{L} \mathbb{P}(R \leq r)dr}.$$

For portfolio optimization (see the recent papers [12, 13]), one can consider the returns $R_1, \ldots, R_n$ for n assets and define the Omega function as $\Omega(x_1R_1 + \cdots + x_nR_n; L)$, where $x_1, \ldots, x_n$ denotes the invested fractions. One can

consider a normalization constraint of the form $x_1 + \cdots + x_n = 1$ and rule out short selling by imposing $x_1, \ldots, x_n \geq 0$.

We applied our derivative-free bilevel method (Algorithm 5.2) to the robust optimization of the Omega function for the example reported in [13] which has 8 financial assets (leading to 7 upper level variables after eliminating $x_8^u = 1 - x_1^u - \cdots - x_7^u$). We considered only one robust parameter (the threshold L), which is the single lower level variable $x^\ell = L$ (along with the lower level constraint $0 \leq x^\ell \leq 0.04$). The specific problem we tried to solve can be formulated as

$$\begin{aligned} \min_{(x^u, x^\ell) \in \mathbb{R}^7 \times \mathbb{R}} \quad & -\Omega(x^u; x^\ell) \\ \text{s.t.} \quad & x_1^u + \cdots + x_7^u \leq 1, \quad 0 \leq x_i^u \leq 0.75, \ i = 1, \ldots, 7, \\ & x^\ell \in \arg\min \left\{ \Omega(x^u; z^\ell) : \ 0 \leq z^\ell \leq 0.04 \right\}, \end{aligned}$$

where

$$\Omega(x^u; x^\ell) \ = \ \Omega(x_1^u R_1 + \cdots + x_7^u R_7 + (1 - x_1^u - \cdots - x_7^u) R_8; x^\ell).$$

The performance of the algorithm on this example is described in Figure 4. Optimization without derivatives of portfolio selection problems involving the Omega function will be the subject of a separate study. Note that the problem stated above is a simplification of more complex models since typically one would like to maximize the Omega function subject to a given maximum level of risk or other constraints (more complex than just limiting the contribution of each asset as we did above by setting $0 \leq x_i^u \leq 0.75, i = 1, \ldots, 7$).

8. Conclusions

This paper represents a first contribution to the derivative-free numerical solution of bilevel optimization problems. We chose to solve both optimization levels by trust-region interpolation based methods, using a similar core algorithm where the use of minimum Frobenius norm quadratic models plays a relevant role. We have taken advantage of the structure of bilevel optimization given by the existence of a lower, inner optimization problem to explore ways of saving objective function evaluations.

One technique that appears to be promising is to solve the lower level inexactly, tightening the accuracy of its solution as the upper level iterations evolve. This idea is related to truncated or inexact Newton and SQP methods, which has proved to be both effective and robust. Another technique we studied and tested consisted of reusing points, for the lower level

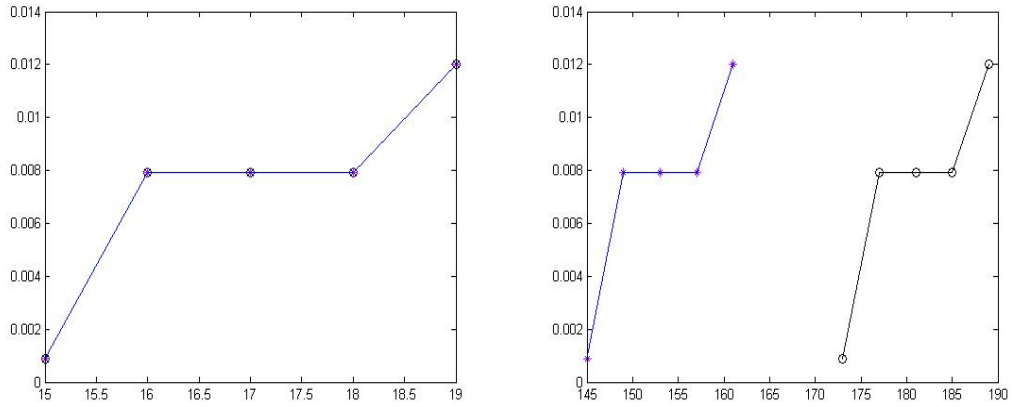

FIGURE 4. Result of the application of Algorithm 5.2 to the maximization of the Omega function from [13] (8 assets), robustly with respect to the return level/threshold. The plots show the value of the reduced upper level function in terms of the number of upper level function evaluations (left) and lower level function evaluations (right). (See the caption of Figure 1 for information about the curves.)

optimization, that have been previously evaluated, even if they correspond to different, slightly perturbed values of the upper level variables. This approach has also shown some capability of reducing the overall effort. However it does not appear to be as robust as the previous idea, and we recommend it only when the lower level function evaluations are relatively expensive.

The overall number of lower level function evaluations required to solve the problems we tested (those reported in this paper and others) can still be considered high. However, per lower level solution this number is relatively low even when considering the most efficient existing trust-region interpolation-based codes for derivative-free optimization.

Finally, we showed how our bilevel approach can be used to solve robust derivative-free optimization problems, and we applied it to the robust maximization of the Omega function in portfolio optimization. One possible difficulty that arises in this approach is that when the (upper level) objective function is convex, the lower level problem results in the minimization of a concave function. This did not apparently give rise to undue difficulty in our limited numerical experiments. Instead, the problem seems rather to lie in the nonconvexity of the *reduced* upper level function.

An astute reader will notice that there are no comparisons with competing methods — this is because such methods appear not to exist.

Acknowledgments. We thank Katya Scheinberg for suggestions regarding Algorithm 5.1 (parts of Steps 5 and 7).

References

[1] J. F. Bard, *Practical Bilevel Optimization: Algorithms and Applications*, Kluwer Academic Publishers, Dordrecht, 1998.

[2] D. Bertsimas, O. Nohadani, and K. M. Teo, *Robust optimization for unconstrained simulation-based problems*, Oper. Res., 58 (2010), pp. 161–178.

[3] A. Cascon, C. Keating, and W. Shadwick, *The Omega function*, tech. report, The Finance Development Centre, London, 2002.

[4] L. M. Case, *An ℓ_1 Penalty Function Approach to the Nonlinear Bilevel Programming Problem*, PhD thesis, University of Waterloo, Canada, 1997.

[5] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *Trust-Region Methods*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2000.

[6] A. R. Conn, K. Scheinberg, and Ph. L. Toint, *On the convergence of derivative-free methods for unconstrained optimization*, in Approximation Theory and Optimization, Tributes to M. J. D. Powell, M. D. Buhmann and A. Iserles, eds., Cambridge University Press, Cambridge, 1997, pp. 83–108.

[7] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Math. Program., 111 (2008), pp. 141–172.

[8] ——, *Global convergence of general derivative-free trust-region algorithms to first and second order critical points*, SIAM J. Optim., 20 (2009), pp. 387–415.

[9] ——, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2009.

[10] S. Dempe, *Foundations of Bilevel Programming*, Kluwer Academic Publishers, Dordrecht, 2002.

[11] G. Fasano, J. L. Morales, and J. Nocedal, *On the geometry phase in model-based algorithms for derivative-free optimization*, Optim. Methods Softw., 24 (2009), pp. 145–154.

[12] S. J. Kane, M. C. Bartholomew-Biggs, M. Cross, and M. Dewar, *Optimizing Omega*, J. Global Optim., 45 (2009), pp. 153–167.

[13] B. Minsky, M. Obradovic, Q. Tang, and R. Thapar, *Applying a global optimisation algorithm to fund of hedge funds portfolio optimisation*. 2009.

[14] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[15] M. J. D. Powell, *Least Frobenius norm updating of quadratic models that satisfy interpolation conditions*, Math. Program., 100 (2004), pp. 183–215.

[16] K. Shimizu, Y. Ishizuka, and J. F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer Academic Publishers, Boston, 1997.

A. R. Conn
Department of Mathematical Sciences, IBM T.J. Watson Research Center, Route 134, P.O. Box 218, Yorktown Heights, New York 10598, USA (arconn@us.ibm.com).

L. N. Vicente
CMUC, Department of Mathematics, University of Coimbra, 3001-454 Coimbra, Portugal
(lnv@mat.uc.pt).